Tampereen yliopisto

Sanna Salonen

EVALUATION OF UI COMPONENT LIBRARIES IN REACT DEVELOPMENT

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiner: Terhi Kilamo
Examiner: Antti Sand
April 2023

TIIVISTELMÄ

Sanna Salonen: Evaluation of UI Component Libraries in React Development
Diplomityö
Tampereen yliopisto
Tietotekniikka
Huhtikuu 2023

UI komponentti kirjastojen tarjoamia valmiita komponentteja käytetään front-end kehitystyön virtaviivaistamisessa. Menetelmän hyötyihin kuuluu säästetty aika ja raha, joka voidaan komponenttien kirjoittamisen sijasta käyttää uusien ominaisuuksien toteuttamiseen ja sovelluksen laadun varmistamiseen. Työn tavoitteena oli selvittää, onko olemassa kriteerejä, joilla voisi arvioida kirjastoja ja vertailla niiden sopivuutta erilaisiin projekteihin.

Arviointikriteerit rajattiin viiteen kategoriaan. Komponentit ja resurssit käsittelivät kirjaston tarjonnan kattavuutta ja monipuolisuutta. Dokumentaatio käsitteli tekijöitä, joilla voi määritellä dokumentaation laadukkuuden. Yhteensopivuus käsitteli kirjaston käytettävyyttä erilaisissa käyttökonteksteissa, kuten eri alustoilla, laitteilla ja käyttäjillä. Rajoitukset käsitteli, mitä yleisiä ja teknisiä esteitä kirjaston käytölle voi olla. Luotettavuus käsitteli, miten aktiivinen kehitys ja yhteisö vaikuttavat kirjaston pitkän aikavälin luotettavuuteen.

Työn arviointikriteerit testattiin kolmella eri kirjastolla, jotka olivat React Bootstrap, MUI ja Ant Design. Työn lopputulos oli, että kriteerit onnistuivat selvittämään kirjastojen välille selkeitä eroja niiden hyvissä ja huonoissa puolissa. React Bootstrap osoittautui hyväksi kirjastoksi pienille projekteille, MUI on erittäin monipuolinen vaihtoehto keskitason projekteille ja Ant Design on kaikista paras korkean tason yritysprojekteille. Vaikka omat mieltymykset ja tottumukset yleensä vaikuttavat eniten kirjaston valintaan, on siis olemassa muitakin tekijöitä, jotka voi ottaa huomioon.

Avainsanat: UI, component library, React, React Bootstrap, MUI, Ant Design

ABSTRACT

Sanna Salonen: Evaluation of UI Component Libraries in React Development
Master of Science Thesis
Tampere University
Information Technology
April 2023

UI component libraries in applications are used to streamline the front-end development process by providing pre-built components. The adoption of these libraries results in saved time and costs, giving the developer the chance to focus on implementing more features and improving the application's overall quality. The thesis aimed to discover if there are criteria that can be used to evaluate these libraries and find the most suitable option for each project.

The evaluation criteria were narrowed down to 5 categories. Components and resources addressed the selection and versatility of the features the library provided. Documentation studied the factors which make up high-quality documentation. Compatibility addressed library's usability in various use contexts, such as with different platforms, devices and users. Restrictions addressed general and technical limitations the library may have. Reliability addressed how active development and community can increase library's long-term reliability.

The thesis applied the criteria to 3 UI component libraries: React Bootstrap, MUI and Ant Design. What it discovered was that the criteria effectively differentiated the studied libraries from each other and defined them with clear pros and cons. React Bootstrap proved to be a decent small project alternative, MUI was a highly versatile option for mid-range projects and Ant Design was most suitable for advanced enterprise applications. While personal preferences and familiarity most often factor into the selected library, there are other important factors that should be considered before making the choice for a project.

Keywords: UI, component library, React, React Bootstrap, MUI, Ant Design

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AntD	Ant Design (UI component library)
API	Application Program Interface
ARIA	Accessible Rich Internet Applications
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSX	JavaScript Syntax Extension
MPA	Multi Page Application
MUI	Material UI (UI component library)
SEO	Search Engine Optimization
SPA	Single Page Application
SSR	Server-Side Rendering
UI	User Interface
UX	User Experience
WCAG	Web Content Accessibility Guidelines

1. INTRODUCTION

Most applications, especially on the web, have a UI (User Interface) that presents the actions and data available to the user in an intuitive and usable way. The frontend of the application that includes the UI is therefore an integral part of almost any software development project. However, as numerous different use cases, contexts, and users exist, it becomes challenging to create a UI that is functional in all these situations. Also, the more complex the UI is, the more time and resources it takes to implement and test. As these costs can become significant, many tools have been created to streamline the development process.

UI component libraries were created to provide developers with ready to use components and tools, which drastically reduce amount of work needed to implement a UI. Not only does using a library reduce costs, but this saved time can go into giving developers more time to focus on more important matters, such as implementing new features for the application. A library generally provides a positive effect on the application's overall quality by providing the developer with reusable and easily customizable code. However, as numerous different UI component libraries are currently available, it may be difficult for the developer to pick the best one for their own application.

The purpose of this thesis is to evaluate some of the most popular UI component libraries available in React. The research question is: what criteria can be used to evaluate the suitability of a UI component library? These criteria will first be discussed and then applied in a case study that utilizes some of the most popular and promising libraries currently available.

In the second chapter of this thesis necessary background information to understand web frontend development is introduced. In the third chapter criteria used to evaluate React UI component libraries is discussed and the libraries chosen for the case study are presented. In the fourth chapter the evaluation criteria are applied to a case study that utilizes the chosen libraries. Finally, in the fifth chapter conclusions are drawn from the results of the case study.

2. WEB FRONT END ARCHITECTURE

Web applications refer to the kind of remote server applications that are accessed via a device's web browser while being connected to the internet. Regardless of the device, the user can instantaneously access the application without needing to install or update it to their own device. Native applications typically have better performance and more features than web applications, but the ease of access for web applications is unparalleled. Additionally, progressive web applications can offer the user an experience on par with native applications by using various web-platform features [1].

Web applications consist of a front end and a back end, but the definition of the specific line between them can vary from application to application. For the purposes of this thesis, front end is defined as the parts of the application the user directly interacts with, such as the UI and the associated code that runs within the web browser, while back end consists of the data and logic that resides within a server, which the front end communicates with via an API (Application Program Interface). Figure 1 showcases the communication flow of a typical web application.

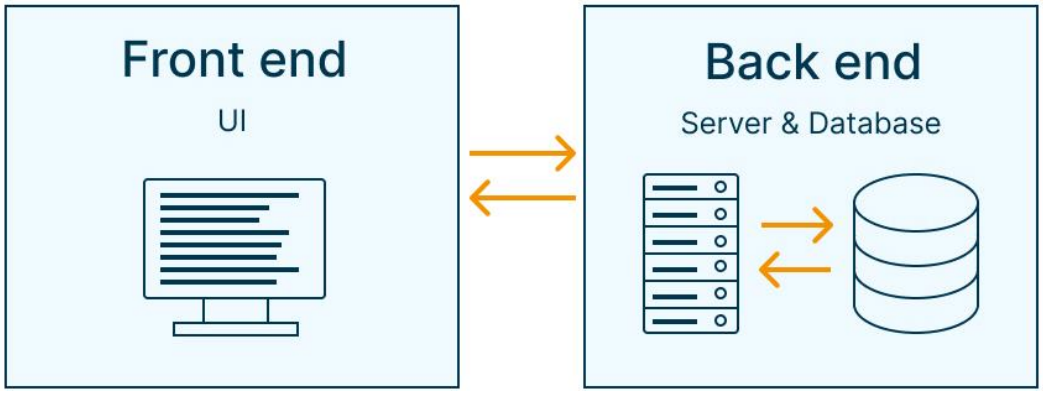

Figure 1. *Web application structure.*

Front end's job is to not only display the data received from the back end, but to also provide the user with an abstraction of the API that is used to mutate the data if necessary. As this code is user facing, the UI and its abstraction of the API has to follow conventions that provide the application with good UX (User Experience).

The following chapter presents some of the basic technologies used in the creation of a web application's front end. As the browser is the common environment for all web applications to run in, the applications share a lot of the same major characteristics.

2.1 HTML

HTML (HyperText Markup Language) is used to build the meaning and structure of the UI before appearance and functionality is added to it with other technologies [2]. HTML forms the basic foundation of the UI and structures the raw data received from the back-end API call in a more presentable format.

Anything that is displayed in the UI is an HTML element. These elements describe the various types of content that can exist on a web page, such as text, lists, titles, containers, or images. As HTML is a markup language, it annotates the data inline with what is referred to as tags. A tag consist of the element's name and two angle brackets that surround it. If a tag in the HTML document is placed before the data it annotates, its corresponding closing tag is placed right after the data to wrap it. The closing tag is marked with a forward slash. The following example wraps a piece of text inside a paragraph element:

```
<p>Text.</p>
```

As tags can also wrap other tags, together they eventually form a tree-like structure where the tips of the tree’s branches are empty tags or tags containing text. The root of the tree consists of a single <html> tag, which wraps inside itself all the other HTML elements. It is preceded by a <!DOCTYPE> tag, whose job is to ensure the browser renders the document correctly as HTML. Directly under the <html> tag are the <head> tag, which has the metadata, and the<body> tag, which contains any elements visible to the user. Following this specified structure is important in creating a HTML document that is used correctly by the browser and other entities that may interact with it.

The importance of correct structure applies to the elements within the <body> tag as well. Semantic HTML refers to HTML elements that have meaning, such as <form>, <table> and <article>. Non-semantic elements, such as or , are simply wrappers, telling nothing about the data they contain.[3] As HTML’s job is to annotate data, non-semantic elements should be avoided if there exists a semantic element that can be used instead. When using semantic elements, they sometimes must be applied in correct sequences so that the document as a whole has meaning. For example, when using a <ul> tag to annotate an unordered list, its children should consist of <li> tagged elements, representing each of the individual items in the list.

Using semantic HTML has various benefits, such as influencing how search engines find the document's keywords as well as making the code more legible for developers and screen readers [4]. Additionally, different HTML elements typically have distinct default

styles applied to them when rendered by the browser. This ensures that even without custom styling, the rendered HTML document is legible to the human eye.

The HTML tags themselves can be customized with the selection of attributes each element offers. Values can be given to attributes by writing their names within the starting tag. The attributes allow additional information, functionality, or styles to be connected to the element. The following example displays an image with an <img> tag that has src and alt attributes:

```
<img src=”images/cat.png” alt=”Cat” />
```

The src attribute contains the path for the image being displayed and the alt attribute contains the description meant for screen readers. This is why aside from semantic HTML attributes are an important way to support accessibility in an application. For example, ARIA (Accessible Rich Internet Applications) properties can be used as a way to provide information to screen readers when non-semantic HTML elements are used [5].

2.2 CSS

CSS (Cascading Style Sheets) is a style sheet language that is standardized according to W3C specifications across all browsers [6]. It is used to define how an HTML document is rendered, giving the document a distinct style that differentiates itself from the default styles set by the browser. CSS should always be used for stylizing instead of picking the HTML tags to use purely based on their default appearance while not adhering to semantic HTML. CSS is versatile and has the capability to modify fonts, colors, layout, positions, sizes, animations, visibility and much more.

CSS is applied to the HTML typically in two ways: either by applying a CSS style sheet to the HTML document via the <style> element or using it inline with the style attribute. The former method is typically used when the styles affect multiple HTML elements at once, while the latter method is used when the styles should affect only a specific elements and possibly its children. Where the CSS style sheet is applied to affects its scope.

The style sheet is a collection of rulesets, each ruleset typically consisting of a selector and 1 or more property-value pairs. The following example features a selector for all <p> tags with the property “color” and value “red”:

```
p {
    color: red;
}
```

The property-value pair defines the style or behavior of the HTML element, CSS supporting more than 200 different properties by default [7]. All properties cannot be used

for all the HTML elements, but they provide a high degree of flexibility and freedom when it comes to the styling.

So that CSS can differentiate between all the different HTML tags, a selector is used to specify which tags the ruleset applies to. The universal * selector applies the styles to every tag in the scope of the style sheet. If more specificity is required, the selector can be used to target only specific tags. Multiple different selectors can be chained together as well to combine them into various patterns, such as nesting or logical operators.

The element selector uses the HTML element's name as the selector and applies the styles to all the tags of that element. Any attributes given to HTML tags can be used as selectors as well. The class attribute is most typically reserved for styling purposes, each tag being able to have one or more classes that can be shared between other tags as well. Program 1 utilizes the class attribute for applying CSS to the HTML document:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <style>
5.       .custom-text {
6.         font-size: 20px;
7.         color: #FF0000;
8.       }
9.     </style>
10.  </head>
11.  <body>
12.    <p>Unstyled text.</p>
13.    <p class="custom-text">Text.</p>
14.  </body>
15. </html>
```

Program 1. *HTML document with CSS applied to it with the class attribute.*

Some HTML tags also come pre-equipped with pseudo-classes that tell what the component's state is, e.g. if the user is hovering over the tag with their cursor, the pseudo-class :hover becomes available. Pseudo-classes provide the CSS with conditions where some styles become active only in certain cases.

2.3 JavaScript

JavaScript is "a lightweight, interpreted, or just-in-time compiled programming language with first-class functions" [8]. JavaScript's purpose in the browser is to add functionality to the HTML document, making it interactive and dynamic.

JavaScript was originally designed specifically for web development and is still used to this day. JavaScript is the primary programming language used in web browsers, ranking as 7th on the TIOBE index in April 2023 [9]. The language commonly used in browsers is actually known as ECMAScript (ECMA-262), a standardized version of Netscape's

JavaScript and Microsoft's JScript created by ECMA International. ECMAScript is currently on its 13th edition published in June 2022.[10] However, JavaScript is not the only programming language available to web browsers despite its dominant position. WebAssembly, for example, can supplement JavaScript by providing a way to run other programming languages alongside it with a near-native performance [11].

One of the main characteristics of JavaScript is its loose typing. JavaScript can be used to store values inside variables without specifying what kind of data it is. While this approach makes the code more versatile, it also makes debugging much more difficult as the types are checked only during runtime. TypeScript, a superset of JavaScript, was developed as an alternative for developers who prefer strong typing. TypeScript checks types of variables during compile time, highlighting any possible errors in the editor before the code is run [12].

JavaScript's primary job in the browser is to make the UI interactive by providing a way for the otherwise static UI, built out of HTML and CSS, to become dynamic. JavaScript code can be applied to the HTML document, for example, with the <script> element. It can be used to handle various events within the browser with event listeners. Many of the HTML elements have interactive attributes that can be connected to JavaScript code, such as onClick, which registers clicks on the specific element. The JavaScript code that executes as a result of an event can also be used to interact with various APIs.

One of the inbuilt APIs available to JavaScript in browser environments is the DOM (Document Object Model) API. When the client requests a webpage, the received HTML document is represented as a DOM. Using its API makes it possible to manipulate HTML and CSS during runtime, changing the page's structure and styles dynamically. For example, program 2 uses a script to by display a text once a button is clicked:

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <script>
5.       function handleClick() {
6.         document.getElementById("hello").innerHTML = "Hello world!"
7.       }
8.     </script>
9.   </head>
10.  <body>
11.    <button type="button" onclick="handleClick()">Say hello</button>
12.    <p id="hello"></p>
13.  </body>
14. </html>
```

Program 2. *HTML document with a script.*

Non-browser APIs can also be interacted with, and these include the application's own back end. HTTP (Hypertext Transfer Protocol) requests to the back end, and other various methods like event handlers, can take a long time to execute and are thus done with asynchronous functions. This ensures that the application remains responsive to events instead of waiting for the long running task to be finished and freezing before executing any other code.

2.4 Frameworks and libraries

HTML, CSS and JavaScript form the 3 basic building blocks of a web application’s front end, but on their own they can be very slow to adapt to advanced use cases. It may be hard to follow good programming practices, or the developer may need to write even the most basic features from scratch. The code may as a result become bloated, unnecessarily complex, poorly tested, and difficult to maintain. This is because HTML, CSS and JavaScript on their own force the developer to focus on minor details that take time away from the bigger picture. Due to this issue, higher-level abstractions are needed to shift the developer’s focus and to improve efficiency. Frameworks are collections of higher-level code, whose purpose is to make development faster and easier. They not only make the development process more efficient, but also increase the general maintainability and quality of the application.

Web front-end frameworks tend to come in two types: JavaScript and CSS frameworks. JavaScript frameworks, as the name indicates, affect the writing of JavaScript within the application. They typically provide various functions that encourage a certain kind of programming style and structuring of the application's code. These functions also provide various highly customizable features, which would be laborious to implement from scratch in plain JavaScript. Some of the most popular JavaScript frameworks for web front-end development include React, Angular and Vue. While JavaScript frameworks focus on functionality, CSS frameworks focus on styling. A CSS framework is a collection of CSS style sheets that can be quickly applied to HTML with the use of classes and other attributes. This saves time when creating components and layouts, because all the CSS does not need to be written from scratch.

The term library is often used when referring to frameworks and vice versa, but they have an important distinction. Framework, as its name suggests, gives the application structure by providing a basic foundation, which the rest of application is built on. Most of the code provided by a framework is versatile and needs high degrees of customization from the developer to make it perform its intended job. A library, on the other hand, is more focused on providing the developer with pre-written code and ready to use components.

These can be directly embedded to the application with very little amount of work, saving the developer from always having to reinvent the wheel. Naturally, some frameworks also have some qualities of libraries and vice versa, so the distinction between them is not always so clear. Frameworks and libraries should also not be confused with other front-end technologies, such as CSS preprocessors, as those are more akin to extensions of the languages themselves.

2.5 React

React is an open-source JavaScript framework developed by Meta [13]. Applications built with React are SPAs (Single Page Applications): "web app implementation that loads only a single web document, and then updates the body content of that single document via JavaScript APIs" [14]. SPAs use JavaScript within the client's browser to render a page's contents while more traditional MPAs (Multi Page Applications) render each page on the server-side. SPAs typically load much more quickly but may struggle with a particularly slow initial render times and issues such as SEO (Search Engine Optimization). React is used to create SPAs, but it also supports SSR (Server-Side Rendering) where the initial page is rendered on the server, but the subsequent renders are done on the client-side with JavaScript. This improves on performance and SEO. There also exists NextJs, which is an alternative JavaScript framework that enables React-based applications to utilize SSR more extensively [15].

React utilizes a virtual DOM, which increases performance by minimizing the amount and scope of re-renders needed to the actual DOM when the page contents are updated. The idealized virtual DOM is kept in memory and manipulated in a declarative manner, having its state changes synchronized with the actual DOM only when necessary.[16] This method is very efficient compared to the traditional way of modifying the actual DOM directly. For the developer, this makes the code straightforward to write and read as the code represents the idealized virtual DOM where there's no need to keep track of all the dynamic manipulations to the actual DOM.

React is based around components, the code being divided into reusable modules, which are combined to create the desired UI. Like typically with SPAs, there is one root document to which the all the components are attached to in a tree-like structure, rendered conditionally based on the visible page and the state of the application. Each component is declared either as a function or a variable that returns HTML at the end. Function components can have attributes just like default HTML elements can, only being referred to as props in React. Program 3 showcases how React function components are used:

```
1. function Hello(name) {
2.   return <p>Hello, {name}!</p>;
3. }
4. const root = ReactDOM.createRoot(document.getElementById(“root”);
5. const hello = <Hello name=”World” />;
6. root.render(hello);
```

Program 3. *React application rendering one function component.*

JSX (JavaScript Syntax Extension) turns the otherwise static HTML dynamic with the help of JavaScript. If a variable is wrapped in curly braces, the virtual DOM is kept up-to-date with its value.[17] This approach also supports conditional rendering where the element can be mounted in and unmounted from the DOM, as showcased in program 4:

```
1. function Conditional(name) {
2.   return (
3.     <div>
4.       <h1>Header</h1>
5.       {name && <p>{name}</p>}
6.     </div>
7.   );
8. }
```

Program 4. *React component where a <p> element is rendered conditionally.*

Often variables in React are stored in the component's state where they exist during the entirety of the component's lifecycle and are be updated as seen fit. The state is initialized as the component mounts and reset as it finally unmounts, meaning the state doesn't persist beyond the component's lifecycle. The state in React function components is typically managed with the useState hook. The initialized state is stored in a variable that can be updated with its associated function for setting the state:

```
const [name, setName] = useState(“initial name”);
```

Beyond useState, React offers various other hooks, such as useEffect, useCallback, useMemo and more. Hooks offer various methods that help with the management of state and lifecycles within a React function component.

3. REACT UI COMPONENT LIBRARY

Effective React development is dependent on reusable, customizable and modular code, so all of that code does not need to be written from scratch. The development process can be significantly streamlined with the help of a UI component library, which provides ready-to-use components that are commonly found in UIs. These components are customizable collections of HTML, CSS and JavaScript that can be imported into the React project from an installable module. Many of these libraries don't include only components either, but also offer various helpful development tools.

To find an answer to the research question of “what criteria can be used to evaluate the suitability of a UI component library”, the following chapter goes through criteria that can be used to evaluate UI component libraries. These criteria will be based on personal experience and be supported by literature. Examples of some popular libraries, that will be tested in a case study in chapter 4, are also presented.

3.1 Evaluation criteria

As the selection of UI component libraries available for use is vast, it may be hard to choose a suitable one between them for your own project. Many developers have personal preferences, but the purpose of the following chapter is to present a set of criteria that can be used to evaluate the pros and cons of libraries in a more systematic manner. The following chapter goes through not only what defines a UI component library, but also presents some potential ways to evaluate them, separated by 5 categories divided into their own subchapters. These criteria aim to differentiate the libraries from each other and determine if there even is a major difference, outside of personal preferences, between the potential choices.

3.1.1 Components and resources

An integral part of an UI component library is its selection of components that provide functionality commonly found in most modern websites. Besides these React components, UI component libraries also provide other functions, hooks and even non-React resources that exist to speed up development and support the components.

Some elements with component-like qualities already exist in plain HTML, such as buttons and inputs, but they require extensive customization. Their default styling is what the designs of most modern websites would rather avoid in favor of creating a more

distinct and user-friendly experience. HTML on its own not only suffers from looking generic, but its looks may also vary from browser to browser, requiring additional styling for consistency [18]. This is a problem, because having consistency is important in creating a predictable and holistic experience for the users of the application [19]. Many common UI components are also much more specialized, not available as plain HTML elements. Without using a UI component library, the developer is forced to create many of these components from scratch. Even CSS frameworks rely on the developer knowing how to correctly utilize its CSS classes to create the components they need, resulting in a workflow that is not only slower but also harder to learn compared to using a component library. The developer may also try to scan through other sources for code to copy and paste into the project, but this comes with the risk of errors and repeating old mistakes [19]. This is not only inefficient, slow, and risky, but it also makes the application more complex. This can lead to maintainability issues, which is why a library can instead be used to help with clean coding and upkeep of the application [20]. A library is often the most reliable way to acquire components not found from plain HTML.

As HTML's selection is quite limited, one way to measure the value of an UI component library is in what variety of components it provides. Components can, for example, be classified into four main categories of containers, controls, widgets, and interaction design patterns, as well as more specific subcategories [21]. If prioritizing scalability and the ability to adapt to even more complex and large-scale projects, the selection of components should cover as many functionalities as possible. The point of a UI component library is to avoid forcing the developer to build components from scratch and this is not possible if the selection is not large enough for the application’s purposes. However, a library should not just be judged purely based on the amount and variety of components it provides. Instead, it should first be considered what kind of components are needed in the given application and then if the library provides anything that is suitable for that purpose. Some applications may only need some specific components and as a result a lightweight library with relatively little amount of features is enough. Heavyweight libraries can have the downsides of bloat, poor performance and a steep learning curve.

How large the variety of components the library has to provide can also be tied to how extensively customization is supported. The developer should have the capability to turn the component into what they need if a specific functionality is not already found from the selection. While simple functionalities may not require extensive customization capabilities, having the components be flexible to various purposes is a great benefit. It ensures that anything the developer needs can be achieved with their library of choice.

A component is customized in two ways: functionality and styling. Functionality is typically customized through the component's API while the styles can also be customized with CSS.

One way to customize a UI component library's styles, while keeping everything consistent, is using a theme. The theme provides the one truth for the palette, typographies, component styles, modes and much more. It is typically implemented with design tokens by having the code reference values that are pulled from a JSON file. As the library's components have very specialized APIs, the theme directly integrates with them. This makes using the library's own theme more efficient than the conventional method of using global CSS files. Thanks to this integration it is also possible to find prebuilt themes that can be quickly applied to the application. If the application needs its own unique theme, many libraries also provide tools that can be used to design one. However, as themes are usually only used to apply general styles that impact multiple components at once, it is beneficial to have more localized ways of applying CSS as well. These CSS tools are typically reserved for individual cases where a customization shouldn't affect every other component in the application. As the provider of UI components, the library should ensure that their style customization is not only fast but also convenient.

Style customization is typically easier if the base component is simple and unstyled, but extreme customization may not always be desirable. Sometimes what the developer needs is a component they can use as is without any tweaking, especially if the goal is fast prototyping. Therefore, the functionality that the developer does not need to be concerned with should be hidden behind the API. This is preferable to making the developer implement everything themselves, because it makes the component easier to use and the code less prone to bugs. Likewise, the components should come pre-equipped with styles. The components provided by a UI component library are generally made to follow design standards that have undergone testing over the course of many years. As a result, the components are not only consistently stylized, but their usability is also very high. As what matters more to users than flashy visuals is functional behavior, the components being ready-to-use without customization is a great benefit.[18] What the developer gets is an efficient development process and a high-quality application. However, one downside of relying on a design system without customizing it is that it can make your application look similar to other applications using the same UI component library. That's why support for customization is important.

There is potentially a choice to be made between customizability and the component being ready to use. If the application is required to have a very distinct design, the component library should allow for high amounts of customization and be simple by default.

However, if this not a great concern, for example in a situation where a visual designer cannot be consulted, the library should be something that is easy and fast to use as is.[18] As the developer may not have qualifications for UX and UI design, it may be better to rely on the library's default settings. The amount and variety of components needed also depends on this choice. The question of what kind of components are needed and how much the library's selection needs to be customized to achieve the desired result. A general all-purpose library, therefore, tends to not only offer a wide selection of different ready-to-use components, but it also has them equipped with high levels of customizability. However, if the design requirements are specific and clear enough, a more specific library is often enough and perhaps even preferable.

Component library can offer various resources as well. As icons are used by most applications, the components may have a specific set of icons meant for their design system. Using the library's own icons ensures that they can be easily and quickly integrated into the components while still looking consistent with the rest of the application's design. Besides tools meant for development, a component library can also offer tools to support the design process as well. These range from theme creators to UI kits, which can be used within design software. A UI component library being associated with a design system with well-defined guidelines is a great benefit from the perspective of designers.

3.1.2 Documentation

The main purpose of a UI component library is to cut down on the amount of work required to create a functional application by using prebuilt components and features. Ideally, using a library should be faster and easier than creating components from scratch, but this is not possible if the developers do not understand the library they're working with. The visibility and clarity of a library's features is achieved with good documentation and by reading it the developers should quickly understand how to use the library to its full potential. Good documentation also reduces the amount of time needed for training, making it easier for a developer to start working with a brand-new library they have no prior experience with.

The library can provide this with a coherent API that clearly describes what features are available to get the component up and running. This not only supports faster development, but also the ability to quickly get the basic functionality working for prototyping and testing purposes. Convoluted and complex APIs that are hard to use should be avoided, if possible. Beyond basic functionality, even more advanced features should be provided in a way that is as easy as possible to understand.

The library should also provide more detailed descriptions of the components. These are included in the documentation that is typically provided on the library's primary website. With the documentation the user can not only understand how everything works, but also easily discover what features are available to them. If the developer, due to poor documentation, does not realize a feature already exists, they may waste their time on manually creating it from scratch. In addition to descriptions of APIs, good documentation often provides various guides and code samples. These code samples are useful in speeding up the developer's work by being possible to copy and by teaching them the basics of how the component works in practice. Finally, the documentation as a whole should be coherently structured so the information the developer requires is easy to find.

3.1.3 Compatibility

It is ideal that the application implemented with the chosen library is functional in as many different use contexts as possible so as not to exclude any possible users. These use contexts include differences in platforms, devices, and users. Compatibility issues can take a lot of testing to tackle, which is why UI component libraries are developed with these issues in mind.

Firstly, it may be difficult to write CSS that works on all browsers and platforms without looking different. UI component libraries typically solve this issue by having already been tested on various browsers, saving the developer from this burden. Customizing the components may result in issues, but if used as is, they should be guaranteed to work as intended regardless of the browser. Cross-browser compatibility with the latest versions of most popular browsers is therefore an important quality for a library to have.

Secondly, responsive design should come with the components, ensuring that they are adaptable to all screen sizes and devices. Using the application on any platform should feel seamless.[18] The library should, therefore, not only provide responsive components, but offer the developer ways to customize it as seen fit.

Finally, the components should be accessible, adhering to standards such as WCAG (Web Content Accessibility Guidelines) [22]. The components should follow semantic HTML by default and also offer ways to customize the HTML elements if necessary. All the other ways to add in accessibility, such as ARIA properties, should be possible through the component APIs. This will ensure the application is compatible with a wide array of different users, including those who may use keyboard navigation, screen readers or other assistive technologies. If the components include text within themselves, the library should provide a way to implement localization.

3.1.4 Restrictions

The usage of any kind of library comes with its own set of restrictions. Before choosing a library, it’s important to check whether it can be utilized in the desired use context or if some restrictions make it less desirable.

Firstly, the weight of the library can in certain cases be a restriction as a large bundle size may negatively affect loading times and performance. While this may not be major a concern for enterprise applications, low bundle size is better for more general applications, especially ones with mobile device users.[23] If the application does not require all the features of a heavyweight library, a more lightweight option is generally preferable as it is not associated with clutter and performance issues [20]. When picking the library, it should be determined which features are necessary and which can be left out.

Secondly, some projects may not be able to utilize a specific library if the support for the desired technologies is not available. This issue can apply to older versions of the technology as well since interoperability needs to be kept up to date. If wanting to use a CSS preprocessor, you may need to consider its interoperability with the library and if such support is needed [20]. Other React frameworks and libraries may need to be interoperable as well, such as the link components offered by routing libraries. Whether the library supports TypeScript is also relevant as it is a commonly used superset of JavaScript. Finally, while this thesis focuses on React libraries specifically, it should be considered if the library is available on other JavaScript frameworks as well. This is a matter of interest as developers may want to transfer their knowledge of their library to other projects that don't utilize React.

Finally, licensing affects to what extent the library can be used and for what purposes. Open-source options may, for example, be provided with a MIT license that gives the developer the rights to "use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software" as long as it is associated with a required copyright notice [24]. However, many other license options exist, such as Apache License 2.0 [25], GNU license [26] and more, each having different conditions. Not all libraries are open source either and some have a hybrid model where certain features are free and some need to be paid for. What the use context is and how much the project is willing to pay for the library should be addressed.

3.1.5 Reliability

As using a third-party library involves relying on a tool developed by someone else, it comes with risks concerning reliability. These risks should be mitigated or avoided if possible.

The library should ideally be actively maintained and developed. This ensures not only that any existing issues will be fixed, but also that the library will continue to be improved in the future with new features. As other technologies also continue to develop, the library needs to be updated regularly to ensure its interoperability with them. Some libraries can in worst cases become deprecated and discontinued, which can make future maintainability of the applications utilizing them challenging. That's why when choosing a library, it should be considered who the contributors to the project are. With an open-source library, the developer community contributes to the development of the library themselves, helping with its maintenance. The more popular a UI component library is, the more active the community is in helping to improve it.

Additionally, an active developer community means that finding support and answers to questions on sites like Stack Overflow is easier. While good documentation can answer a lot of questions, it can't answer all of them. Popularity of a library has another added benefit from the point of view of users. A component library that they've seen elsewhere feels more reliable to them as they have previous experience of it. Thus UI component libraries, that have been already utilized by popular websites many users are familiar with, will increase the application's usability [18].

3.2 Libraries

The case study in chapter 4 will utilize some UI component libraries as an example and evaluate them using the criteria defined in chapter 3.1. The scope of this thesis narrowed the selection down to 3 entries. According to the reliability criteria, libraries with high popularity and development activity in recent years are preferable. Therefore, deprecated and discontinued libraries were not considered in the evaluation.

Three primary metrics were used to measure the activity of the developer community: Stack Overflow questions, GitHub stars and GitHub contributors. Libraries that performed consistently well compared to their peers on these metrics were chosen. The data was taken from Stack Overflow Trends and is represented in figure 2.

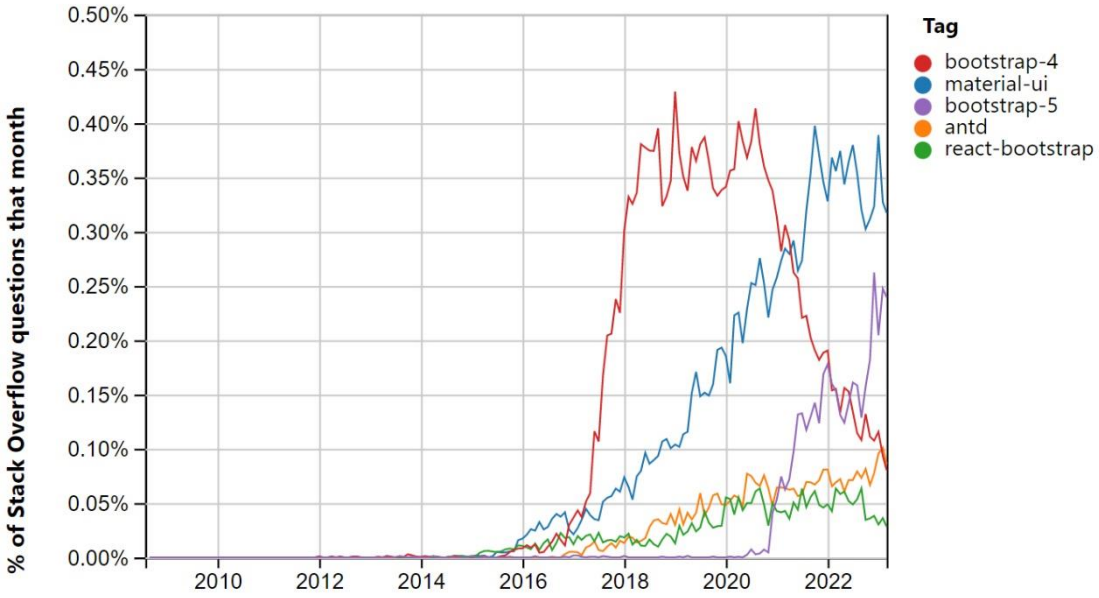

Figure 2. *Stack Overflow Trends for Bootstrap, MUI, and Ant Design [27].*

3.2.1 React Bootstrap

React Bootstrap is a replacement for Bootstrap JavaScript, the most popular front-end framework, having all its components rebuilt from scratch in React. Among the things React Bootstrap claims is it being accessible by default and compatible with thousands of already existing Bootstrap themes.[28] React Bootstrap's website is showcased in figure 3.

Figure 3. *React Bootstrap website [28].*

The first chosen library was React Bootstrap, because Bootstrap JavaScript has been greatly popular over the years. Bootstrap JavaScript's Stack Overflow questions amount to 33,8k in total [29, 30]. Even Bootstrap v5 has a rising trend in questions, showing how it is a valid replacement for v4. In addition to that, Bootstrap JavaScript has 163,0k stars

and 1,4k contributors on GitHub [31]. However, it is important to note that React Bootstrap is much less popular than its predecessor, having 4,0k questions on Stack Overflow as well as 21,6k stars and 0.5k contributors on GitHub [32, 33]. For the purposes of the case study, it will be examined how the most popular front-end framework's React version holds up on its own despite its lower popularity.

3.2.2 MUI

MUI, formerly known as Material UI, is a fully-loaded component library based on Google's Material Design. MUI claims that its benefits include timeless aesthetics, intuitive customization, unrivaled documentation, and dedication to accessibility.[34] MUI's website is showcased in figure 4.

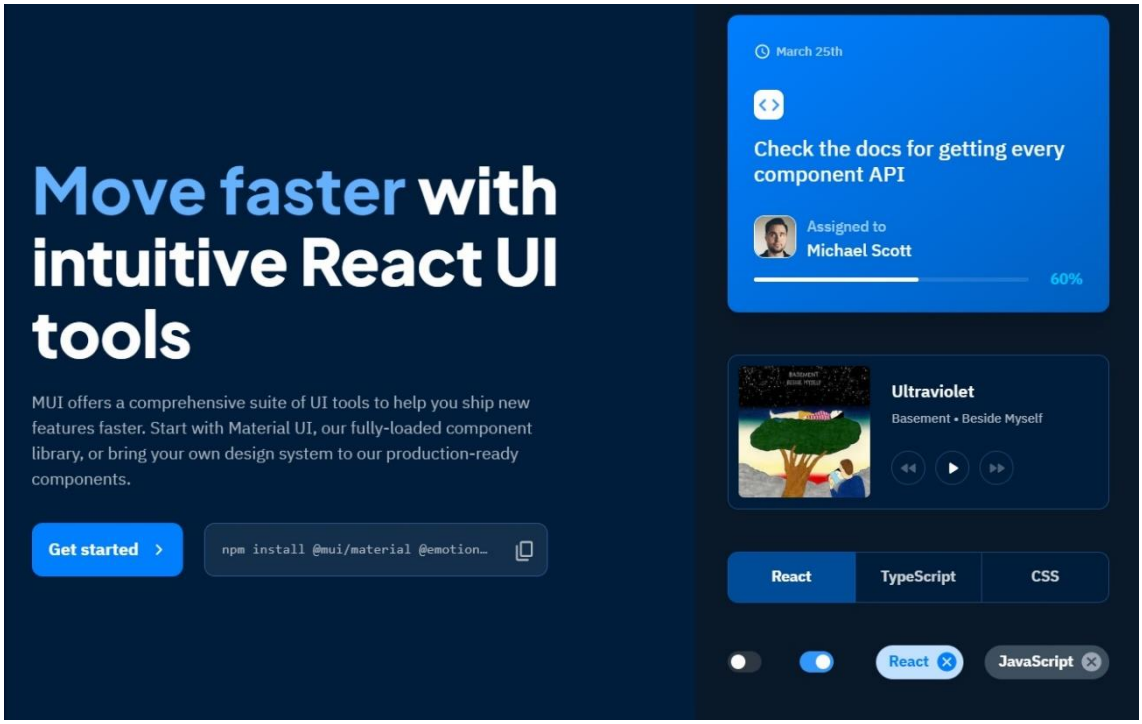

Figure 4. *MUI website [34].*

MUI was chosen for the evaluation as it has been steadily gaining popularity over the years and become the most popular UI component library specifically meant for React. It has 21,2k questions on Stack Overflow as well as 85,7k stars and 2,7k contributors on GitHub [35, 36]. The library is actively developed and maintained and has the potential to surpass even Bootstrap JavaScript in terms of popularity.

3.2.3 Ant Design

Ant Design, also referred to as AntD, is a design system developed by the Alibaba Group, offering component libraries for JavaScript frameworks such as React [37]. AntD's website is showcased in figure 5.

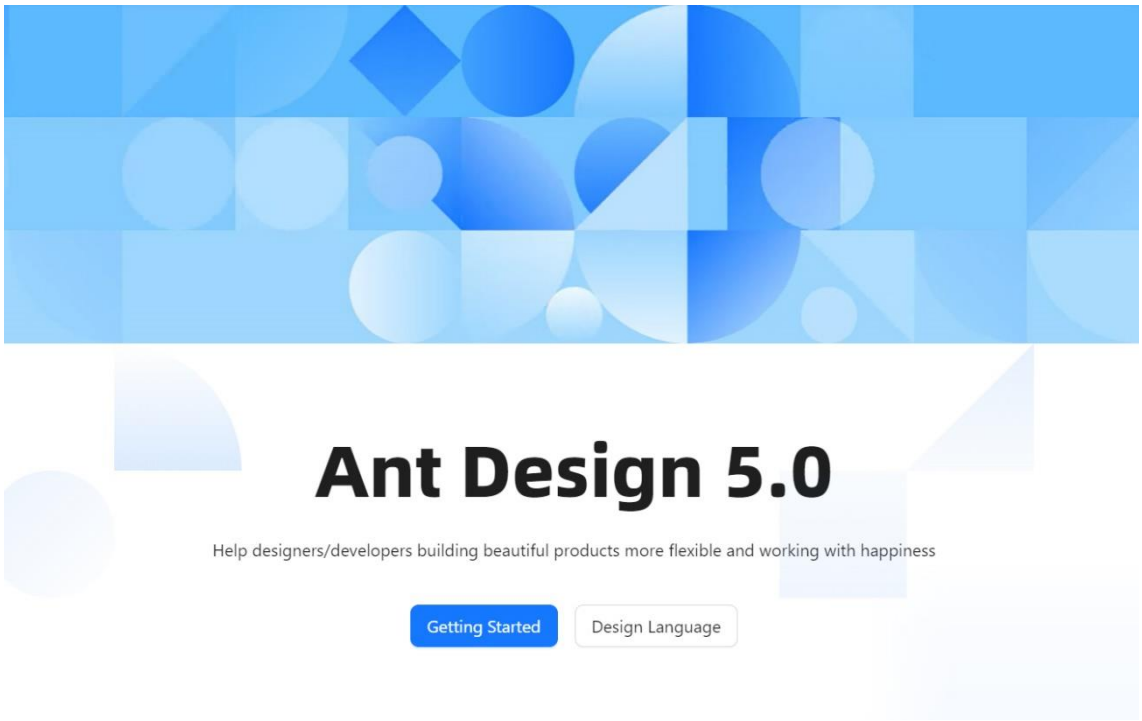

Figure 5. *Ant Design website [37].*

AntD was chosen for the evaluation thanks to its consistent popularity. It has a total of 5,0k questions on Stack Overflow as well as 85,3k stars and 1,9k contributors on GitHub [38, 39]. While lagging behind in Stack Overflow questions to MUI and Bootstrap JavaScript, the trend is slowly rising as seen in figure 1. AntD also has almost as many stars as MUI on GitHub.

4. CASE STUDY

The following chapter applies the evaluation criteria discovered in chapter 3.1 to the UI component libraries chosen in chapter 3.2. The goal is to not only determine the validity of the criteria by applying them to a concrete example, but to also to see if the most popular UI component libraries even have any notable differences between them. The original categories discovered in chapter 3.1 were divided into a total of 13 criteria, which are presented in table 1.

Category	Title	Description
Components and resources	Amount	What is the number of components provided in total?
	Variety	What different kinds of components and features are provided?
	Customizability	How ready to use the components are and what kind of customization features for both styling and functionality exist?
	Design tools	What tools are provided for the UI and UX designers specifically?
Documentation	Documentation	What is the quality of the documentation's structure and contents?
Compatibility	Cross-browser compatibility	Is the library usable on all platforms?
	Responsiveness	Is the library usable on all devices and screen sizes?

	Accessibility	Is the library usable for all users?
Restrictions	Bundle size	How large is the library's bundle size?
	Interoperability and support	What support exists for other technologies, libraries and frameworks?
	Licensing	What licenses does the library offer and how much do they cost?
Reliability	Active development	Is the library being actively developed?
	Developer community	Is the community surrounding the library active?

Table 1. *UI component library evaluation criteria.*

4.1 Evaluation

The evaluation criteria presented in table 1 are gone over one by one and applied to each of the libraries. As the reliability criteria were already used to narrow down the choice of libraries for the case study in chapter 3.2, they will not be addressed in the following chapter.

4.1.1 Amount

React Bootstrap provides 32 unique components. One of them is a Form component that can be used to create various form inputs such as selects and checkboxes, making the total component count offered by React Bootstrap higher than 32 in practice.[40] More generic low-level versions of the components are available in Restart UI, which is what React Bootstrap is largely based on [41]. Additionally, Bootstrap Icons is a selection of 1900+ icons that can be used with or without Bootstrap in any project [42].

MUI provides 61 unique components, 5 of which are not part of MUI Core features. [43]. 2 of them are part of MUI X, which supports more advanced and complex functionalities [44]. 3 are Lab components, which are not ready to be included in MUI core yet [45]. In

total, the component API documentation describes 138 different components when some of the 61 components described are broken into smaller parts. MUI also provides style variations of the core components. 18 unstyled variations are included in the MUI Base package, providing a way to fully customize the components without having to override existing designs [46]. MUI also has a sister library, Joy UI, which provides a subset of 32 core components with designs that don't follow Material Design specifications [47]. In addition to components, MUI also provides various utilities, which are not included in the component count of 61. As MUI is based on Google's Material Design, it also provides a selection of 2100+ Material Icons as an installable package [48].

AntD provides 72 unique components in total, 6 of which are not part of core AntD and come with the ProComponents package [49]. These are template components that provide higher level abstractions of the base components [50]. Core AntD also has several related libraries that are advertised alongside it, for example Ant Motion and AntV, all with their own selection of components.

There also exists AntV, which provides the necessary components for creating data visualizations [51]. Finally, AntD provides a set of 780+ downloadable icons [52].

React Bootstrap provides the smallest number of components compared to its peers, but its selection is still comprehensive enough for most basic applications. Naturally, some even more lightweight alternatives, that are not discussed in this case study, exist. MUI and AntD are both relatively heavyweight libraries that support more advanced applications. AntD especially has a vast selection of components and is thus the largest of the libraries. It is important to note that all the components listed above do not come by default with MUI's core 61 and AntD's core 72 components and are instead available as various optional packages. These optional packages include each library's set of icons as well, of which MUI has the widest selection of them and AntD has the smallest. However, it is always possible to use external icons with all the libraries, even if they may not integrate as well with the components and the design system. Overall, React Bootstrap has the benefits of a lightweight library and is easy to get started with due to not having too many components, but MUI or AntD especially are needed if you're building a more complex application.

4.1.2 Variety

React Bootstrap provides 2 primary layout components, 2 utility components, a form component and 27 other types of components, which are not classified into any specific categories.[40] Despite this lack of specific classification, the 27 components cover some common functionalities an application could have. Low-level components from Restart

UI offer unstyled variants of the React Bootstrap components. Most Bootstrap icons are divided into 2 variants: outlined and filled. Additionally, Bootstrap themes can not only provide custom styles but also higher-level templates. However, as React Bootstrap is a complete reimplementation of Bootstrap JavaScript, it may mean some themes that include templates are not automatically compatible, as they can extend default JavaScript behaviors [53].

MUI classifies its components to 7 categories (excluding MUI X and Lab): inputs, data displays, feedback, surfaces, navigation, layout and utils [43]. They cover numerous different functionalities that could be commonly found from an application, not concentrating on only specific kinds. Different style variations of the same components are provided by Joy UI and MUI Base, latter of which is completely unstyled. Material Icons are provided in 5 variants: filled, outlined, rounded, two tone and sharp [48]. In addition to icons, various higher-level templates created from MUI's own components are provided in the MUI store to further speed up development process [54].

AntD classifies its components in 7 categories: general, layout, navigation, data entry, data display, feedback and other. There is also a separate category for ProComponents.[49] Outside of core AntD, great variety is provided with the vast selection of associated packages. Even further variety is added to the selection with the inclusion of AntV, which provides vast selection of graphs that can satisfy most needs. AntD also has its own animation package called Ant Motion, which integrates with AntD components [55]. AntD's own set of icons come in 3 variants: outlined, filled and two tone [52]. Finally, AntD's higher-level templates are provided in their own package called Ant Design Pro, a production-ready solution for admin interfaces [56]. Ant Design Landing and Scaffolds also offer a wide variety of different templates [57, 58].

Due to its smaller amount of components, React Bootstrap offers less variety than MUI and AntD, even if the components offered do cover some of the most common functionalities. For more specific purposes customization or another component library is required. MUI, for example, has a large selection of all types of components available. However, AntD in particular stands out with its large selection of advanced components, which are for built for the most complex applications. AntV data visualization is something neither React Bootstrap nor MUI offer, and it directly integrates with the rest of AntD library, creating a seamless experience in the application as opposed to using a 3rd party graph library. While React Bootstrap and MUI are good enough for most applications, AntD's variety is large enough to satisfy even enterprise needs. Additionally, all the libraries offer templates. Bootstrap primarily relies on community themes, some of which

will not work with React Bootstrap due to compatibility issues, even if the collection despite this is still quite large thanks to Bootstrap's popularity. MUI and AntD's own selection of templates, on the other hand, are guaranteed to always work. However, AntD stands out from its competition again by not only offering more templates, but also by offering template-like higher-level components with Ant Design Pro. Combined with Ant Design Landing and Scaffolds it is more versatile and varied than Bootstrap or MUI's templates. However, while AntD's variety is greater, even MUI is enough for most applications.

4.1.3 Customizability

React Bootstrap provides all Bootstrap components written from scratch in React, making them not only ready to use, but also customizable with CSS or the prop API. To help with writing CSS, Bootstrap JavaScript provides style sheets and Sass integration, which are compatible with React Bootstrap as well [53]. Sass is a CSS extension and preprocessor that reduces complexity of CSS style sheets [59]. This means that components can be styled with props and classes that follow the Bootstrap JavaScript naming patterns. All Bootstrap libraries also come with themes: bundles of HTML, CSS and JavaScript, which include prebuilt styles, layouts and components that can be readily used in an application. React Bootstrap relies on Bootstrap style sheet and is thus compatible with thousands of Bootstrap themes available [28]. Themes or Bootstrap variables can be further customized with custom Sass files [53]. However, as established, some extensions of default JavaScript behavior may not work with React Bootstrap, so if such a theme is desired, Bootstrap JavaScript may have to be used instead. Generally a theme is guaranteed to work in a React Bootstrap project as long as it sticks to default classes and variants [60]. If wanting to use the unstyled components provided in Restart for easier customization instead, they will be treated as a separate library.

MUI components are based on Google's Material Design system and thus ready to be used as is, both visually and functionally. Even MUI Base, which provides unstyled components, comes with production-ready functionality [46]. Despite this MUI claims its components are still flexible, giving the developer full control over their look and behavior [34]. While Material Design provides an appealing aesthetic, it's very distinct to Google and can thus require customization to make application feel more unique. MUI components can be customized with the prop API or via CSS. All MUI components have their individual HTML elements adorned with CSS classes, which can be easily targeted to override the base styling. All these classes are listed in each component's API documentation. They either describe the element's type or its state, similarly to pseudo-classes. Existing CSS can easily be overridden, for example, with MUI's own sx prop, Global CSS

overrides, styled utilities, or the theme.[61] The theme ensures that the entire application has a consistent look and feel [62]. Themes can also be nested and placed more locally within the HTML tree to limit their scope, but in those cases another methods of styling should be used instead. Emotion is MUI's style engine that is used to generate CSS styles for the components [63]. It is best used for individual components to create clear and modular code. If very extensive customization is required, MUI Base's unstyled components don't require the developer to override the existing Material Design styling [46].

AntD also has its components based on a design system of the same name, providing production-ready components. Functionality can be extended or modified with the prop API. Style customization in AntD is primarily done with themes. AntD 5.0 themes utilize CSS-in-JS, which makes it possible to "use JavaScript to describe styles in a conflict-free and reusable way" [64]. The scope of the theme affects can be controlled with component tokens, telling the theme which components to modify and which to ignore. Themes can also be nested and placed more locally within the HTML tree so that their styles only affect a localized part of the application. Additionally, themes can be quickly generated with the help of algorithms that are provided with version 5.0. This adds a lot of versatility to AntD theme styling, although it can also add to complexity of the structure.

All 3 libraries provide versatile styling methods. As their themes' structures are standardized, themes can be custom-made or readily acquired from external sources. They also provide a quicker way to customize components than plain CSS. Notably, React Bootstrap's themes double as templates, which MUI and AntD provide separately. Outside of themes, MUI provides the most versatile styling options thanks to Emotion coming with its default installation. As Emotion is built to directly integrate with MUI, using it is very convenient and it produces clean code. React Bootstrap on the other hand has the benefit of being related to Bootstrap JavaScript, integrating with it and taking advantage of its high popularity when finding themes and support. AntD's theming is quite versatile as well, making it possible to easily modify the styles of components. However, it has less prebuilt themes on offer.

4.1.4 Design tools

Bootstrap components can be used in design software if a suitable community version of them exists, because no official UI kit has been published by Bootstrap. Bootstrap is not a design system, having originated as a CSS framework. As a result it has less defined style guidelines than UI component libraries directly based on existing design systems. However, Bootstrap still has icons are that are available as SVG files that can be downloaded and used in design purposes [65].

MUI comes with a selection of UI kits built for popular design software such as Figma, Adobe XD, and Sketch [66]. Material icons are also downloadable in SVG format. Being based on Google's Material Design, it has very well-defined style guidelines that can be utilized in the design process. Numerous 3rd party services can be used in the design process of MUI applications as well. For example, there exists a Material Design palette generator for picking colors for a theme [67]. Any prior knowledge of Material Design guidelines is therefore an advantage when designing MUI applications.

AntD offers a wide range of different design resources, such as editing tools, plugins and UI kits for design software like Figma, Adobe XD and Sketch [68]. Ant Design Landing even offers its own editor for designing UIs with various AntD templates on offer [57]. As the UI component library is based on a design system of the same name, very well-defined design guidelines are available. Like with Material Design, knowledge of Ant Design guidelines is a benefit when designing AntD applications.

Overall, MUI and AntD provide various official design tools and guidelines while Bootstrap relies more on community-built tools. AntD especially has the largest collection of official tools provided on its website and these cover most needs a designer may have. Whether something is marked as official on the library's website or not doesn't necessarily affect its quality, but it may make it more difficult to see what tools are at one's disposal. However, popular libraries, like the examples, are at an advantage as their high scores in terms of reliability criteria ensures support for various development and design tools. It is likely that if less popular libraries were included in this case study, they would have less design tools on offer.

4.1.5 Documentation

React Bootstrap's documentation is compact and mostly offers some basic technical information to get developers started. Most importantly, it offers code examples as well as descriptions of the components and their API. In comparison, Bootstrap JavaScript's documentation is much more extensive and offers more useful information about the library such as about accessibility, customization, utilities and more. While some of this information applies to React Bootstrap as well, there are many cases where the similarities and differences remain unclear. Sometimes the documentation states whether or not something applies to both libraries while sometimes there is no mention of it at all. When using React Bootstrap, the documentation offers only the bare minimum and you may need to rely on external sources for further information.

MUI claims that its documentation is unrivaled with a total of 2000 contributors [34]. Noteworthy is that the official documentation of MUI is quickly accessible from the developer's

code editor. In Visual Studio Code, hovering over the component's HTML tag provides a direct link to the component's documentation on the official website, saving time from having to find the right page in the browser. The documentation on the website is divided by package, but the underlying chapters are shared by most of them. These chapters offer technical information, guides, design resources, code examples, component APIs and available CSS classes. The interactive code examples can also be viewed and edited through CodeSandbox or Stackblitz. The documentation offers such a sizable amount of content that it may even be hard to locate what you're looking. Despite this size, Material Design's documentation can't be accessed on MUI's own website.

AntD's documentation has a clear division of content on its website, the documentation being divided into design, development, components, blog, and resources. All the necessary materials for both designers and developers are quickly accessible from the top navigation. Design section has a detailed overview of the design system the library is based on, development includes technical information and guides. Components includes descriptions of the APIs as well as interactive code examples, which can be viewed or edited through CodeSandbox, Stackblitz or CodePen. Resources includes links to all the available design tools, articles, and useful design references. Finally, blog includes updates and stories. The information is clearly categorized so it is easy to find what you're looking for despite the documentation's length. The footer of the website includes links to the documentation of other packages related to the core React library, such as Ant Design Pro and AntV. However, the documentation of some of these packages lacks English translations. The core AntD documentation is fully in English, but the others have sections, or the entirety of them, only in Chinese, such as with Ant Design Mini. AntD's GitHub issues and community support are also often in Chinese. The language barrier poses a problem if the developer does not speak the language.

Overall, MUI offers the largest amount of information within its documentation. In particular, it has the most guides and general advice for problem situations. AntD's documentation is not far behind, benefitting from a clear and comprehensible structure, but the language barrier can become problematic. React Bootstrap's documentation in comparison to MUI and AntD is very short and its relationship with Bootstrap JavaScript's documentation is confusing. In addition, the documentation also lacks design resources in comparison to the other two libraries, focusing mostly on development.

4.1.6 Cross-browser compatibility

React Bootstrap claims to support all browsers that are also supported by Bootstrap JavaScript and React [53]. Bootstrap JavaScript supports the latest, stable releases of all

major browsers and platforms, stating that even in alternative browsers it should, in most cases, function correctly [69].

MUI is also designed to work on the latest stable releases of all major browsers and platforms, which include Edge, Firefox, Chrome, Safari and more. Internet Explorer is partially supported but this support will be dropped in MUI v6.[70]

AntD is designed to support the last 2 versions of modern browsers and downgrade processing is available for legacy compatibility. This, in addition to browser support, makes AntD more compatible with other CSS frameworks.[71]

As the UI component libraries chosen for the case study were chosen based on popularity and high reliability, they are all kept up-to-date with the latest releases of major browsers. Some less popular libraries may not be kept to the same standard, but it would require further studies to confirm. However, even the most popular libraries could not guarantee that some less popular browsers are supported. AntD offers the highest amount of legacy compatibility, but it still requires manual customization. It is a possibility that users of less popular browsers may not be able to use an application utilizing these libraries as intended.

4.1.7 Responsiveness

React Bootstrap uses Grid and Stack components to create its layouts. The breakpoints can be customized when using Bootstrap's source Sass files within the theme provider component.[72] These Breakpoints are accessible outside of React Bootstrap components with a selection of special React hooks.

MUI, as its primary layout components, provides Grid, Stack, Box and Container. MUI's own breakpoints can be customized in the theme and the scope can range from global to local based on where the theme providers are located [73]. The breakpoints can be accessed within MUI's sx prop, styled components or the useMediaQuery React hook, which can be used with regular CSS media queries as well.

AntD's own breakpoints can be customized with the theme as well, either globally or locally. The Grid and Layout components provide a responsive system with the breakpoints, but the breakpoints are accessible from the prop API as well. There also exists a hook useBreakPoint for utilizing the breakpoints within other code.[74]

The responsiveness tools the three libraries provide are roughly the same, consisting of breakpoints defined within the theme as well as various ways to access them from within the code, such as from the prop API or with hooks. MUI provides an advantage by having

the styled engine integrate with the theme, providing a way to utilize the theme's breakpoints within React code.

4.1.8 Accessibility

React Bootstrap claims that each component is accessible by default and built with it in mind [28]. This means that the components are designed to work on keyboard navigation, ARIA labels and more. React Bootstrap's component documentation often describes how accessibility is, or can be, implemented. While React Bootstrap components are accessible, it is important to remember that if Bootstrap JavaScript is used as well, its accessibility depends heavily on the developer having a correct implementation. However, it should always be possible to adhere to WCAG and similar standards with the library.[75] Localization does not come built in with React Bootstrap, but the text labels of the components can be switched with the developer's own localization library of choice.

MUI states accessibility is its dedication and that it is one of their highest priorities when shipping new features [34]. If accessibility is not set by default within the component, the documentation describes how the developer can make a component adhere to accessibility standards. For example, Autocomplete component's documentation encourages the usage of WAI-ARIA label for the textbox [76]. Material Design in general is strict about defining the accessibility guidelines for each component. MUI also provides 100 of the most common locales for the text contents within its components, not requiring the developer to write the locales themselves. However, for less common locales and customizations, it is possible to use your own locale file as well.[77] Finally, MUI's "component" prop, or "variant" prop within Typography components, can be used to determine what HTML element the component is rendered as, making it possible to easily write semantic HTML.

AntD, unlike React Bootstrap and MUI, does not make any special note of its adherence to accessibility standards. The component documentation doesn't describe if components are accessible and also lacks specific instructions for making them accessible if the support is not built in by default. This is due to the design system, unlike in Material Design, not dictating how accessibility is implemented. However, most components still are keyboard navigable and support WAI-ARIA labels. In general, using AntD requires the developer to pay close attention to accessibility and take more responsibility for its implementation. Despite its accessibility issues, AntD provides 67 locales for the text contents of its components and supports custom locales as well [78].

Historically, all the libraries have had some accessibility issues, but most of those issues have been taken care of as the libraries have received extensive development over the years. Thanks to being open source, potential issues are quickly pointed out and fixed. With AntD, the developer may have to take more responsibility when handling accessibility due to the lack of guidance within the documentation. In comparison, MUI and Material Design have very detailed accessibility guidelines, which makes adhering to accessibility standards easy for the developer. Localization is possible with all libraries, but React Bootstrap does not come with in-built locales, forcing the developer to define the translations themselves. This is less efficient as labels have to be changed manually within each instance of the component rather than by changing only one localization file.

4.1.9 Bundle size

The latest version of React Bootstrap is available as an npm package @react-bootstrap [53], default installation for MUI comes with packages @mui/material, @emotion/react and @emotion/styled [79], and AntD comes installed with the package @antd [80]. Additionally, icons for these libraries are provided in packages @react-bootstrap-icons, @mui/icons-material and @ant-design/icons. This selection includes the basic installations of these libraries and icons, excluding any of the optional packages. The bundle sizes of all the latest versions of these packages are described in table 2.

Package	Bundle size, minified + gzipped (kB)
@react-bootstrap	37.4
@react-bootstrap-icons	322
@mui/material	137.1
@emotion/react	7.9
@emotion/styled	4.9
@mui/icons-material	686.7
@antd	387
@ant-design/icons	158.1

Table 2. *MUI package sizes [81, 82, 83, 84, 85, 86, 87, 88].*

Bootstrap is the most lightweight library of the three, which is also apparent from its lesser amount of components and features. As such it is a suitable library for smaller projects that still require some prebuilt UI components.

MUI is a more heavyweight library due to its sizable amount of features, but its documentation comes with instructions for limiting its bundle size [89]. Some components and features are also divided into separate packages, making it easier to leave out installations, which are not necessary [90]. However, MUI is still quite large and should be used when its features are actually needed in the project.

AntD has by far the largest bundle size of the libraries and as a result offers the largest amount of components and features as well. It is an enterprise-grade library and is thus most wise to use with complex projects that absolutely require the features it has to offer. AntD projects are easy to scale up as the various optional packages can be added to the project when needed, e.g. AntV.

The three libraries are all popular but provide a good range of variance in terms of how heavyweight they are. The application's scope and complexity therefore affects, which one of them is most suitable. Naturally, while lightest choice in this case study, React Bootstrap is also not the most lightweight UI component library currently available, which means that for some simple projects even it may be excessive if sizeable amount of its features are not needed.

4.1.10 Interoperability and support

React Bootstrap is compatible with TypeScript and offers an npm package @types/react-bootstrap with the required type definitions [91]. There also exists integration for other third-party libraries, such as React Router in the form of @react-router-bootstrap package [92]. React Bootstrap is specifically built for React, but Bootstrap has versions of it meant for other JavaScript frameworks. The original Bootstrap JavaScript library is a general-purpose CSS framework, but the more specialized versions of it are optimized for specific JavaScript frameworks, such as BootstrapVue for Vue and ng-bootstrap for Angular.[93] This ensures that Bootstrap can be effectively used regardless of the Javascript framework.

Some technologies MUI v5 supports include, for example, TypeScript, Node.js, routing and style libraries [70, 94, 95]. MUI is specifically implemented for React, but the Material Design system has been used in UI component libraries meant for other JavaScript frameworks, such as Angular Material and Vue Material [96, 97]. Material Design is also available in Flutter [98].

AntD offers a list of 30+ library recommendations to use with AntD, including examples such as custom date libraries, routing libraries, and much more [99, 100]. AntD's CSS compatible adjustments also make some CSS frameworks work better with the library [71]. AntD is not only compatible with TypeScript, but it has been built with it, being based

on its latest stable of version [101]. As for support for other JavaScript frameworks, AntD design system has been implemented in Angular and Vue, although the React version is the official and primary implementation [102]. There also exists Ant Design Mobile, which is a collection of native components specifically built for cross-platform mobile applications [103].

The interoperability and support the three libraries offer is very similar. As they are all popular libraries, it is apparent that in one way or another, support for common technologies and libraries has been implemented due to the high demand and high amount of active developers. AntD has the most extensive documentation regarding the matter, followed by MUI, but lack of documentation doesn't necessarily mean that support for a library or a technology doesn't exist. Being based on design systems, it is possible to find UI component libraries that have implemented it in other JavaScript frameworks and even programming languages as well.

4.1.11 Licensing

React Bootstrap is licensed under MIT, which provides free access to all its main features and components [104]. Bootstrap Themes license comes separately in 3 different distribution variants: standard, multisite, and extended [105]. Each individual theme is paid for once with costs ranging from 39 to 559 dollars [106].

MUI's free community plan is licensed under MIT, providing free access to Material UI, Joy UI, MUI Base and MUI system. MUI promises that anything initially released under the MIT license will remains as such forever. However, a commercial license must be purchased to gain access to the advanced features of MUI X as well as support priority.[107] Pro and Premium plans require a paid subscription, being $15/month and $49/month respectively per each developer [108]. MUI templates come in standard, standard plus and extended licenses, making the prices range from free to 599 dollars. Additionally, the UI kits for Figma, Adobe XD and Sketch all cost between $69/year and $79/year per user.[109]

AntD also has a MIT license that provides full access to its features [110]. All the major packages and templates are free, but some of the design resources may have associated costs. The UI kits for Figma and Adobe XD, for example, have one-time payment costs between 99 and 849 dollars based on the chosen license.

Overall, all libraries are open source and thus provide free access to their main features. Some additional features are behind a paywall, but these most often include less common use cases. When it comes to templates and themes, it is not absolutely necessary to buy one and many of them can be found for free anyway. For MUI, some advanced

components require a payment, which is can be an issue if the application needs them specifically. AntD provides advanced components for free, which may make it a more desirable library in those use cases if you don't want to acquire a commercial license for MUI. The UI kits for both MUI and AntD are paid features, but due to Bootstrap's UI kits being community provided, they may be found for free.

4.2 Discussion

The evaluation was mostly a success as it managed to discover key differences between the libraries in majority of the categories. While some of the differences were not significant, each of the libraries have a distinct identity that can influence their suitability for a specific project, going beyond just personal preferences.

React Bootstrap was a relatively lightweight library with a small learning curve, suitable for small and straightforward projects. Even more lightweight options than it exist, but React Bootstrap provides a decent selection of components that can satisfy common needs. While this React library cannot quite live up to the reputation of its CSS framework predecessor, this library is a good option to consider for anyone already familiar with the CSS framework. With the association comes a good amount of support from the community, even if the React library is much less popular.

MUI is a comprehensive, albeit relatively heavyweight, library with a wide selection of features and components. It is a mid-range library that can be suitable for even larger projects. It has a steeper learning curve than React Bootstrap, but once the developer learns how to use it, it becomes incredibly flexible and versatile. Almost any React project can manage with MUI, explaining its popularity. The extremely active community helps with making developing with it easy and convenient. However, while Material Design provides an appealing aesthetic, it is so distinctly associated with Google that it can make your application feel less unique unless customized.

AntD is the most scalable library of the three, capable of satisfying even enterprise needs with its wide selection of features for both designers and developers. It notably offers a large amount of advanced features with a free license. The design system also has a clean and simple aesthetic, capable of being used in almost any situation. However, AntD is very heavyweight so there are better options than it unless the features it offers are absolutely needed. Its support for accessibility is also relatively poor. While popular, finding support for any issues is more difficult than with MUI or Bootstrap.

Overall, all the libraries occupy a different space, having distinct use cases where they are most suitable for. React Bootstrap and MUI cannot handle most complex projects on

their own and would have to paired with other libraries that may not even synergize with them well. On the other hand, AntD can handle almost anything with ease. However, AntD is not suitable for smaller projects unlike React Bootstrap and MUI. More competition exists between React Bootstrap and MUI, but even they occupy different spaces due to React Bootstrap's heavy association with Bootstrap JavaScript, providing a stronger connection to non-React projects.

5. CONCLUSIONS

This thesis was made to answer the research question: what criteria can be used to evaluate the suitability of a UI component library? The goal was to discover common traits UI component libraries share and use a set of criteria based on them to systematically evaluate libraries' pros and cons. This study was divided in 3 parts: firstly relevant background theory was presented, then the evaluation criteria were introduced and finally the criteria were tested in a case study with real libraries.

The results support the use of these criteria to determine how suitable a UI component library is for a project, because notable differences were found between the 3 libraries used in the case study. The evaluation is possible to do even when unfamiliar with the library, supporting its use in finding suitable candidates before starting a new project. However, it is still important to note that for many people the choice is based on personal biases and familiarity, because once you learn how to use one library, you are likely reluctant to switch to another. This approach also reduces the amount of time spent on learning how to use a new library effectively. Despite this, when planning for future projects, it is worth it to consider if there are some factors at play, which may make the other options more suitable.

As the evaluation criteria were only applied to popular UI component libraries with a relatively large amount of features, the next step would be to diversify the case study. This can be achieved by including a more varied representation of different libraries, most importantly less popular libraries. The evaluation criteria also are not complete and can be expanded with even more criteria for more comprehensive results. Some criteria, such as cross-browser compatibility and responsiveness did not provide many differences with the example libraries and may be replaced if this trends continues even with other libraries. Even if both are extremely important factors for forming a good UI component library, most libraries handle these factors decently well and thus there is not much value in comparing how different libraries have implemented them.

REFERENCES

[1] MDN Web Docs – Progressive web apps (PWAs). URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (Accessed: 7.4.2023)

[2] MDN Web Docs – HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (Accessed: 7.4.2023)

[3] W3Schools – HTML Semantic Elements. URL: https://www.w3schools.com/html/html5_semantic_elements.asp (Accessed: 7.4.2023)

[4] MDN Web Docs – Semantics. URL: https://developer.mozilla.org/en-US/docs/Glossary/Semantics (Accessed: 7.4.2023)

[5] MDN Web Docs – ARIA states and properties. URL: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes (Accessed: 7.4.2023)

[6] MDN Web Docs – CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (Accessed: 7.4.2023)

[7] Dofactory – CSS Properties. URL: https://www.dofactory.com/css/properties (Accessed: 7.4.2023)

[8] MDN Web Docs – JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (Accessed: 7.4.2023)

[9] TIOBE – TIOBE Index for April 2023. URL: https://www.tiobe.com/tiobe-index/ (Accessed: 7.4.2023)

[10] ECMA International – ECMA-262. URL: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/ (Accessed: 7.4.2023)

[11] MDN Web Docs – WebAssembly. URL: https://developer.mozilla.org/en-US/docs/WebAssembly (Accessed: 7.4.2023)

[12] TypeScript home page. URL: https://www.typescriptlang.org/ (Accessed: 7.4.2023)

[13] ReactJs home page. URL: https://react.dev/ (Accessed: 7.4.2023)

[14] MDN Web Docs – SPA (Single-page application). URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA (Accessed: 7.4.2023)

[15] NextJs home page. URL: https://nextjs.org/ (Accessed: 7.4.2023)

[16] React Legacy – Docs > Virtual DOM. URL: https://legacy.reactjs.org/docs/faq-internals.html (Accessed: 7.4.2023)

[17] React – Learn React > Writing markup with JSX. URL: https://react.dev/learn/writing-markup-with-jsx (Accessed: 7.4.2023)

[18] Griffiths, David, and Dawn Griffiths. *React Cookbook*. 1st edition, O'Reilly Media, Inc., 2021.

[19] Curtis, Nathan. *Modular Web Design Creating Reusable Components for User Experience Design*. 1st edition, New Riders, 2010.

[20] Shenoy, Aravind, and Anirudh Prabhu. *CSS Framework Alternatives: Explore Five Lightweight Alternatives to Bootstrap and Foundation with Project Examples*. Apress L. P, 2018.

[21] Romero Gómez, Rosa, et al. *"Taxonomy for Rich-User-Interface Components: Towards a Systematic Development of RIAs." Web Engineering*, Springer Berlin Heidelberg, pp. 411–14.

[22] W3C – Understanding WCAG 2.0. URL: https://www.w3.org/TR/UNDERSTANDING-WCAG20/Overview.html (Accessed: 7.4.2023)

[23] Praneeth Reddy, Mukthapuram. *"Analysis of Component Libraries for React JS." IARJSET*, vol. 8, no. 6, 2021, pp. 43–46.

[24] Open Source Initiative – The MIT License. URL: https://opensource.org/license/mit/ (Accessed: 7.4.2023)

[25] The Apache Software Foundation – Apache License, version 2.0. URL: https://www.apache.org/licenses/LICENSE-2.0 (Accessed: 7.4.2023)

[26] GNU Operation System – GNU General Public License. URL: https://www.gnu.org/licenses/gpl-3.0.html (Accessed: 7.4.2023)

[27] Stack Overflow Insights – Trends. URL: https://insights.stackoverflow.com/trends?tags=material-ui%2Cantd%2Creact-bootstrap%2Cbootstrap-4%2Cbootstrap-5 (Accessed: 7.4.2023)

[28] React Bootstrap home page. URL: https://react-bootstrap.github.io/ (Accessed: 7.4.2023)

[29] Stack Overflow – Questions tagged [bootstrap-4]. URL: https://stackoverflow.com/questions/tagged/bootstrap-4 (Accessed: 7.4.2023)

[30] Stack Overflow – Questions tagged [bootstrap-5]. URL: https://stackoverflow.com/questions/tagged/bootstrap-5 (Accessed: 7.4.2023)

[31] GitHub – Bootstrap. URL: https://github.com/twbs/bootstrap (Accessed: 7.4.2023)

[32] Stack Overflow – Questions tagged [react-bootstrap]. URL: https://stackoverflow.com/questions/tagged/react-bootstrap (Accessed: 7.4.2023)

[33] GitHub – React-Bootstrap. URL: https://github.com/react-bootstrap/react-bootstrap (Accessed: 7.4.2023)

[34] MUI home page. URL: https://mui.com/ (Accessed: 7.4.2023)

[35] Stack Overflow – Questions tagged [material-ui]. URL: https://stackoverflow.com/questions/tagged/material-ui (Accessed: 7.4.2023)

[36] GitHub – MUI Core. URL: https://github.com/mui/material-ui (Accessed: 7.4.2023)

[37] Ant Design home page. URL: https://ant.design/ (Accessed: 7.4.2023)

[38] Stack Overflow – Questions tagged [antd]. URL: https://stackoverflow.com/questions/tagged/antd (Accessed: 7.4.2023)

[39] GitHub – Ant Design. URL: https://github.com/ant-design/ant-design (Accessed: 7.4.2023)

[40] React Bootstrap – Components > Alerts. URL: https://react-bootstrap.github.io/components/alerts/ (Accessed: 7.4.2023)

[41] React Bootstrap – Utilities > @restart/ui. URL: https://react-bootstrap.github.io/utilities/restart-ui/ (Accessed: 7.4.2023)

[42] npm – react-bootstrap-icons URL: https://www.npmjs.com/package/react-bootstrap-icons (Accessed: 7.4.2023)

[43] MUI – Components. URL: https://mui.com/components/ (Accessed: 7.4.2023)

[44] MUI – MUI X > Overview. URL: https://mui.com/x/introduction/ (Accessed: 7.4.2023)

[45] MUI – Material-UI > About the lab. URL: https://mui.com/material-ui/about-the-lab/ (Accessed: 7.4.2023)

[46] MUI – MUI Base > Overview. URL: https://mui.com/base/getting-started/overview/ (Accessed: 7.4.2023)

[47] MUI – Joy UI > Overview. URL: https://mui.com/joy-ui/getting-started/overview/ (Accessed: 7.4.2023)

[48] MUI – Material-UI > Material Icons. URL: https://mui.com/material-ui/material-icons/ (Accessed: 7.4.2023)

[49] Ant Design – Components > Overview. URL: https://ant.design/components/overview (Accessed: 7.4.2023)

[50] ProComponents – Quick Start. URL: https://procomponents.ant.design/en-US/docs (Accessed: 7.4.2023)

[51] AntV home page. URL: https://antv.vision/en (Accessed: 7.4.2023)

[52] Ant Design – Components > Icon. URL: https://ant.design/components/icon (Accessed: 7.4.2023)

[53] React Bootstrap – Getting Started > Introduction. URL: https://react-bootstrap.github.io/getting-started/introduction/ (Accessed: 7.4.2023)

[54] MUI – Templates. URL: https://mui.com/templates/ (Accessed: 7.4.2023)

[55] Ant Motion home page. URL: https://motion.ant.design/ (Accessed: 7.4.2023)

[56] Ant Design Pro – Getting Started. URL: https://pro.ant.design/docs/getting-started/ (Accessed: 7.4.2023)

[57] Ant Design Landing home page. URL: https://landing.ant.design/ (Accessed: 7.4.2023)

[58] Ant Design Scaffolds. URL: https://scaffold.ant.design/#/ (Accessed: 7.4.2023)

[59] W3C – Sass Introduction. URL: https://www.w3schools.com/sass/sass_intro.php (Accessed: 7.4.2023)

[60] React Bootstrap – Getting Started > Theming and customizing styles. URL: https://react-bootstrap.github.io/getting-started/theming/ (Accessed: 7.4.2023)

[61] MUI – Material-UI > Customization > How to customize. URL: https://mui.com/material-ui/customization/how-to-customize/ (Accessed: 7.4.2023)

[62] MUI – Material-UI > Customization > Theming. URL: https://mui.com/material-ui/customization/theming/ (Accessed: 7.4.2023)

[63] MUI – Material-UI > Guides > @mui/styled-engine. URL: https://mui.com/material-ui/guides/styled-engine/ (Accessed: 7.4.2023)

[64] GitHub – JSS > Docs > Introduction. URL: https://github.com/cssinjs/jss/blob/master/docs/intro.md (Accessed: 7.4.2023)

[65] Bootstrap – Icons. URL: https://icons.getbootstrap.com/ (Accessed: 7.4.2023)

[66] MUI – MUI in your favorite design tool. URL: https://mui.com/design-kits/ (Accessed: 7.4.2023)

[67] MUI – Material-UI > Customization > Color. URL: https://mui.com/material-ui/customization/color/ (Accessed: 7.4.2023)

[68] Ant Design – Resources. URL: https://ant.design/docs/resources (Accessed: 7.4.2023)

[69] Bootstrap – Getting Started > Browsers and devices. URL: https://getbootstrap.com/docs/5.3/getting-started/browsers-devices/ (Accessed: 7.4.2023)

[70] MUI – Material-UI > Getting Started > Supported platforms. URL: https://mui.com/material-ui/getting-started/supported-platforms/ (Accessed: 7.4.2023)

[71] Ant Design – Development > CSS Compatible. URL: https://ant.design/docs/react/compatible-style (Accessed: 7.4.2023)

[72] Bootstrap – Getting Started > Layout > Breakpoints. URL: https://react-bootstrap.github.io/layout/breakpoints/ (Accessed: 7.4.2023)

[73] MUI – Material-UI > Customization > Breakpoints. URL: https://mui.com/material-ui/customization/breakpoints/ (Accessed: 7.4.2023)

[74] Ant Design – Components > Grid. URL: https://ant.design/components/grid (Accessed: 7.4.2023)

[75] Bootstrap – Getting Started > Accessibility. URL: https://getbootstrap.com/docs/4.0/getting-started/accessibility/ (Accessed: 7.4.2023)

[76] MUI – Material-UI > Autocomplete > Accessibility. URL: https://mui.com/material-ui/react-autocomplete/#accessibility (Accessed: 7.4.2023)

[77] MUI – Material-UI > Guides > Localization. URL: https://mui.com/material-ui/guides/localization/ (Accessed: 7.4.2023)

[78] Ant Design – Development > Internationalization. URL: https://ant.design/docs/react/i18n (Accessed: 7.4.2023)

[79] MUI – Material-UI > Getting Started > Installation. URL: https://mui.com/material-ui/getting-started/installation/ (Accessed: 7.4.2023)

[80] Ant Design – Development > Ant Design of React. URL: https://ant.design/docs/react/introduce (Accessed: 7.4.2023)

[81] Bundlephobia – react-bootstrap@2.7.2 . URL: https://bundlephobia.com/package/react-bootstrap@2.7.2 (Accessed: 7.4.2023)

[82] Bundlephobia – react-bootstrap-icons@1.10.2. URL: https://bundlephobia.com/package/react-bootstrap-icons@1.10.2 (Accessed: 7.4.2023)

[83] Bundlephobia – @mui/material@5.11.16. URL: https://bundlephobia.com/package/@mui/material@5.11.16 (Accessed: 7.4.2023)

[84] Bundlephobia – @emotion/react@11.10.6 . URL: https://bundlephobia.com/package/@emotion/react@11.10.6 (Accessed: 7.4.2023)

[85] Bundlephobia – @emotion/styled@11.10.6. URL: https://bundlephobia.com/package/@emotion/styled@11.10.6 (Accessed: 7.4.2023)

[86] Bundlephobia – @mui/icons-material@5.11.16. URL: https://bundlephobia.com/package/@mui/icons-material@5.11.16 (Accessed: 7.4.2023)

[87] Bundlephobia – antd@5.4.0. URL: https://bundlephobia.com/package/antd@5.4.0 (Accessed: 7.4.2023)

[88] Bundlephobia – @ant-design/icons@5.0.1. URL: https://bundlephobia.com/package/@ant-design/icons@5.0.1 (Accessed: 7.4.2023)

[89] MUI – Material-UI > Guides > Minimizing bundle size. URL: https://mui.com/material-ui/guides/minimizing-bundle-size/ (Accessed: 7.4.2023)

[90] MUI – Material-UI > Guides > Understanding MUI packages. URL: https://mui.com/material-ui/guides/understand-mui-packages/ (Accessed: 7.4.2023)

[91] npm – @types/react-bootstrap. URL: https://www.npmjs.com/package/@types/react-bootstrap (Accessed: 7.4.2023)

[92] npm – react-router-bootstrap. URL: https://www.npmjs.com/package/react-router-bootstrap (Accessed: 7.4.2023)

[93] Bootstrap – Getting Started > JavaScript. URL: https://getbootstrap.com/docs/5.3/getting-started/javascript/ (Accessed: 7.4.2023)

[94] MUI – Material-UI > Guides > Routing libraries. URL: https://mui.com/material-ui/guides/routing/ (Accessed: 7.4.2023)

[95] MUI – Material-UI > Guides > Style library interoperability. URL: https://mui.com/material-ui/guides/interoperability/ (Accessed: 7.4.2023)

[96] Angular Material home page. URL: https://material.angular.io/ (Accessed: 7.4.2023)

[97] Vue Material home page. URL: https://www.creative-tim.com/vuematerial/ (Accessed: 7.4.2023)

[98] Material Design > Develop > Flutter. URL: https://m3.material.io/develop/flutter (Accessed: 7.4.2023)

[99] Ant Design – Development > Third-Party Libraries. URL: https://ant.design/docs/react/recommendation (Accessed: 7.4.2023)

[100] Ant Design – Development > Use custom date library. URL: https://ant.design/docs/react/use-custom-date-library (Accessed: 7.4.2023)

[101] Ant Design – Development > Use in TypeScript. URL: https://ant.design/docs/react/use-in-typescript (Accessed: 7.4.2023)

[102] Ant Design – Design > Introduction. URL: https://ant.design/docs/spec/introduce (Accessed: 7.4.2023)

[103] Ant Design Mobile home page. URL: https://mobile.ant.design/ (Accessed: 7.4.2023)

[104] GitHub – react-bootstrap > LICENSE. URL: https://github.com/react-bootstrap/react-bootstrap/blob/master/LICENSE (Accessed: 7.4.2023)

[105] Bootstrap Themes – Licenses. URL: https://themes.getbootstrap.com/licenses/ (Accessed: 7.4.2023)

[106] Bootstrap Themes – All Themes. URL: https://themes.getbootstrap.com/shop/?orderby=date (Accessed: 7.4.2023)

[107] MUI – MUI X > Licensing. URL: https://mui.com/x/introduction/licensing/ (Accessed: 7.4.2023)

[108] MUI – Pricing. URL: https://mui.com/pricing/ (Accessed: 7.4.2023)

[109] MUI – Store > Populars. URL: https://mui.com/store/#populars (Accessed: 7.4.2023)

[110] GitHub – ant-design > LICENSE. URL: https://github.com/ant-design/ant-design/blob/master/LICENSE (Accessed: 7.4.2023)